Technical Breakdown of Position Based Fluid

Qingyuan Qie
will.qie@mail.utoronto.ca
University of Toronto
Toronto, ON, Canada

Changlin Su
sheldon.su@mail.utoronto.ca
University of Toronto
Toronto, ON, Canada

ABSTRACT

This paper provides a technical breakdown of implementing a parallel version of the Position Based Fluids(PBF) method. We discussed the simulation algorithms of PBF, the effects of different confinements in the algorithms. We also go in depth of the data structures used to implement the algorithms efficiently. Due to the highly parallelizable nature of the algorithm, we provided a implementation using the CUDA platform which achieves realtime simulation with a particle count of 12k. A CPU version is also available.

CCS CONCEPTS

• **Computing methodologies → Physical simulation**.

KEYWORDS

Fluids Simulation, Physics-Based Animation

ACM Reference Format:
Qingyuan Qie and Changlin Su. 2021. Technical Breakdown of Position Based Fluid. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/0.0/0.0

1 INTRODUCTION

Fluids in computer animations are simulated with discrete particles, the surface of the fluids is reconstructed at render time based on the position of the underlying particles. In each time step, the PBF method first predicts the positions and velocity, then corrects the particle positions by enforcing the incompressibility constraints. The new velocity of the particles are computed from the new positions and the position of the particles at the last time step.

2 ENFORCING INCOMPRESSIBILITY

For particle i at position p_i, we compute the density of the fluid around particle i using the estimator:

$$\rho_{F(i)} = \sum_{j \in F(i)} m_j W_{poly6}(\boldsymbol{p}_i - \boldsymbol{p}_j, h) \tag{1}$$

where ρ_0 is the rest density of the fluid, m_j is the mass of the particle j, h is a constant, and W is the Poly6 kernel from [TODO: insert citation]. F is the neighbor function that returns the neighboring particle of particle i.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/0.0/0.0

The Poly6 kernel is defined as follows,

$$W_{poly6}(\boldsymbol{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3 & 0 \le r \le h \\ 0 & \text{otherwise} \end{cases}$$

where r is the norm of $\boldsymbol{r}$. Combine equation (1) and the definition of poly6, we can notice that if the distance between particle i and j is greater than h, particle j does not contribute to the density around particle i. Thus the neighbor-finding algorithm and need to find the particles whose distance to particle j is smaller or equal to h, we will discuss the detail of the neighbor-finding algorithm later.

Then we introduce the constant density constraints C_i. For particle i, we have,

$$C_i(\boldsymbol{p}) = \frac{\rho_i}{\rho_0} - 1 \tag{2}$$

where ρ_0 is a constant that denotes the rest density of the fluid, and $\boldsymbol{p}$ is the position of all the particles in the system.

Since we want the density around each particle is always equal to $\rho 0$, we want to find a particle correction $\Delta(\boldsymbol{p})$ such that,

$$C(\boldsymbol{p} + \Delta(\boldsymbol{p})) = 0 \tag{3}$$

The solution to (3) is found by a series of Newton steps along the constraint gradient

$$\Delta(\boldsymbol{p}) \approx \nabla C(\boldsymbol{p})\lambda \tag{4}$$

$$C(\boldsymbol{p} + \Delta(\boldsymbol{p})) \approx C(\boldsymbol{p}) + \nabla C^T \Delta \boldsymbol{p} = 0 \tag{5}$$

$$\approx C(p) + \nabla C^T \nabla C \lambda = 0 \tag{6}$$

And the gradient of the constraint C_i with respect to a particle k is given by,

$$\nabla_{\boldsymbol{p}_k} C_i = \frac{1}{\rho_0} \sum_j \nabla_{\boldsymbol{k}} W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) \tag{7}$$

Which has two different cases based on whether k is a neighboring particle or k is the particle i itself,

$$\nabla_{\boldsymbol{p}_k} C_i = \frac{1}{\rho_0} \begin{cases} \sum_j \nabla_{\boldsymbol{p}_k} W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) & \text{if } k = i \\ -\nabla_{\boldsymbol{p}_k} W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) & \text{otherwise} \end{cases} \tag{8}$$

The original paper uses the Spiky kernel fot the gradient computation, which is defined as,

$$\nabla W_{spiky}(\boldsymbol{r}, h) = -\frac{45}{64\pi h^6} \begin{cases} (h - r)^2 \hat{\boldsymbol{r}} & 0 \le r \le h \\ 0 & \text{otherwise} \end{cases}$$

where $\hat{r}$ is the normalized $\boldsymbol{r}$.

Plug (8) into (6) and solves for λ yields,

$$\lambda_i = -\frac{C_i(\boldsymbol{p})}{\sum_k |\nabla_{\boldsymbol{p}_k} C_i|^2} \tag{9}$$

Then the position correction $\Delta \boldsymbol{p}_i$ including affect from neighboring particles is

$$\Delta \boldsymbol{p}_i = \frac{1}{\rho_0} \sum_j (\lambda_i + \lambda_j) \nabla W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) \tag{12}$$